Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE **Making Better Use of Bandwidth. Data Compression and Network Managment Technologies**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Rand Corporation,1776 Main Street,PO Box 2138,Santa Monica,CA,90407-2138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**56**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

TECHNICAL
REPORT

Making Better Use of Bandwidth

Data Compression and Network Management Technologies

John F. Pane, Leland Joe

Prepared for the United States Army

Approved for public release; distribution unlimited

RAND ARROYO CENTER

The research described in this report was sponsored by the United States Army under Contract No. DASW01-01-C-0003.

ISBN: 0-8330-3745-5

Published 2005 by the RAND Corporation
1776 Main Street, P.O. Box 2138, Santa Monica, CA 90407-2138
1200 South Hayes Street, Arlington, VA 22202-5050
201 North Craig Street, Suite 202, Pittsburgh, PA 15213-1516
RAND URL: http://www.rand.org/
To order RAND documents or to obtain additional information, contact
Distribution Services: Telephone: (310) 451-7002;
Fax: (310) 451-6915; Email: order@rand.org

Preface

Operations in Afghanistan and Iraq have demonstrated the Army's increasing reliance on communications. Tactical forces on the move and widely dispersed were stressed to communicate voice and other data, including text messages, database transfers, real-time video, and imagery. These applications are bandwidth intensive, an especially challenging problem for forces on the move that cannot use high-gain antennas. The data demands of future forces are expected to increase even further.

This report focuses on how the Army might use bandwidth better, specifically how compression technologies and network management techniques might help to improve effective information throughput.

This research was sponsored by the Army CIO/G-6 and was conducted in RAND Arroyo Center's Force Development and Technology Program. RAND Arroyo Center, part of the RAND Corporation, is a federally funded research and development center sponsored by the United States Army.

For more information on RAND Arroyo Center, contact the Director of Operations (telephone 310-393-0411, ex. 6419; FAX 310-451-6952; e-mail Marcy_Agmon@rand.org), or visit Arroyo's web site at http://www.rand.org/ard/.

Contents

Figures

Tables

Summary

Operations in Afghanistan and Iraq demonstrated the Army's increasing reliance on communications. Tactical forces on the move and widely dispersed were stressed to communicate voice and data and were unable to exchange database transfers, real-time video, and imagery. These applications take a lot of bandwidth,[1] an especially challenging problem for forces on the move that cannot use high-gain antennas. Furthermore, future demands seem likely to increase. Researchers from RAND Arroyo Center have been seeking ways in which the Army might use bandwidth better, specifically how new compression technologies might help improve information throughput. The objective of compression is to reduce the amount of data required to store or transmit digital information.

Compression Techniques

Compression algorithms can exploit several potential methods of data reduction. One is by improving coding efficiency. Coding inefficiency arises when the standard way of encoding a message unit uses more bits than necessary. For example, using a byte (8 bits) to encode 26 alphabetic characters is inefficient because a byte can encode up to 256 distinct characters. A second way to reduce data is to reduce redundancy in messages. A third way is to approximate the message rather than transmit it exactly. For example, video transmission might suppress minor changes from one frame to the next.[2]

Results

Lossless compression algorithms can achieve compression ratios up to 9:1; lossy compression algorithms can achieve ratios as high as 350:1. However, data compression involves tradeoffs. The most important of these is that the aggressive use of lossy compression yields lower quality. To achieve better quality, lossy algorithms are used more conservatively, yielding compression ratios about an order of magnitude lower than their maximums. More complex

[1] Strictly defined, *bandwidth* is the width of the frequency spectrum of a signal, in Hertz. However, this report uses the term's more common meaning: *data rate*, measured in bits per second (bps). These two definitions are interrelated by Shannon's law of information theory, which states that a communications channel of a certain width has a maximum rate at which information can be transferred. This maximum rate is known as the *channel capacity*.

[2] The technique of using an approximation of the original rather than an exact replica is called "lossy" compression. Those algorithms that do not apply this technique are called "lossless" compression. The highest compression rates are typically achieved with lossy algorithms.

algorithms can be designed to retain quality at high compression ratios, at the cost of increased computation. The needs of users may vary over time, so the use of compression techniques needs to remain flexible. Still, the research indicates that the Army can take advantage of commercial compression technologies, although it needs to define what is an acceptable loss in quality and may also have to provide users with increased computing power.

Network accelerators can improve throughput by factors of 2 to 3. Their use requires changes to network structure and operations, but only for the last link to the user. Network accelerators allow the user to control bandwidth usage so that individuals could specify the amount of compression needed for their specific missions. The combination of new compression techniques with network accelerators lets users manage their individual bandwidth needs and could potentially reduce bandwidth demands by an order of magnitude. While most of these technologies are currently commercially available, their use may require research and development into techniques to enhance real-time streaming data as well as user training to manage needs for quality.

Recommendations

Lessons from Operation Iraqi Freedom imply that the Army will need to make best use of available bandwidth. This report discusses how existing data compression and network management techniques could be applied in the near to medium term to improve performance. These techniques would not "solve the bandwidth bottleneck" but would contribute to better performance with minimal impact on existing networks. These techniques are "low-hanging fruit" that could be harvested to increase performance.

We recommend that the Army do the following:

- Incorporate compression and network acceleration technologies into future systems.
- Identify where Army-specific tailoring could improve on commercial data compression technologies.
- Develop an experimental plan to determine acceptable compression-related losses in quality and to train users.

Of these three recommendations, the last one will prove the most difficult to achieve and in some ways is the most important. The recommendation is that the Army develop a systematic experimentation program to determine user needs on an objective basis and, eventually, to train users on what level of communications support they should expect and request. This recommendation is fundamental to determining the required design of future communications networks for the Army.

Acknowledgments

The authors would like to thank Isaac Porche and Jimmie McEver, who served as technical reviewers, for their valuable assistance in improving this report.

Acronyms

ac	Alternating Current
ASCII	American Standard Code for Information Interchange
CECOM	Army Communications and Electronics Command
dc	Direct Current
DCT	Discrete Cosine Transform
DISN	Defense Information System Network
DoD	Department of Defense
DRSN	Defense Red Switch Network
DSN	Defense Switched Network
DVSG	DISN Video Secure Global
DWT	Discrete Wavelet Transform
EMSS	Enhanced Mobile Satellite Services
GIG	Global Information Grid
HDTV	High-Definition Television
JPEG	Joint Photographic Experts Group – image compression scheme
JPEG 2000	Revision to the JPEG image compression scheme
MOSAIC	Multifunctional On-the-Move Secure Adaptive Integrated Communications
MPEG	Motion Picture Experts Group – video compression scheme
NIPRNET	Non-Secure Internet Protocol Router Network
NTSC	National Television System Committee
OEF	Operation Enduring Freedom
OIF	Operation Iraqi Freedom
RGB	Red, Green, Blue – encoding for color images

SIPRNET	Secure Internet Protocol Router Network
STANAG	NATO Standardization Agreement
VTC	Videoteleconferencing
YIQ	Luminance, In-phase, Quadrature – encoding for color images

CHAPTER ONE

Lessons from Recent Operations on Army Demands for Bandwidth

Background

Previous RAND Arroyo Center research identified a number of ways to increase bandwidth, that is, the capacity of communication infrastructure to transmit data (Joe and Porche, 2004). No single technique was found to dramatically increase bandwidth; rather, a combination of techniques was recommended. Some of them will require major changes to the Army's communications networking, particularly moving to higher frequencies, developing new networking technologies and protocols, and using vertical nodes to increase connectivity and throughput.

The Army and Department of Defense (DoD) are developing new communications systems to increase bandwidth from the strategic to tactical levels. These programs include the Transformational Communications Architecture (TCA) to develop a space-based very-high-bandwidth backbone, the Warfighting Information Network–Tactical (WIN-T) for higher tactical echelons, and the Joint Tactical Radio System (JTRS) for lower tactical echelons. These programs will greatly increase both bandwidth and on-the-move capabilities. While initial operating capabilities are planned for 2006, it will take several more years for widespread deployment to occur.

This report focuses on near- and mid-term solutions using data compression, which can be applied in existing communications systems as well as the new ones under development. This strategy for better use of bandwidth may increase throughput for certain types of information by up to an order of magnitude. While these are not sufficient to solve the bandwidth shortfall, they can help to alleviate the near-term problem.

Although the Army has had in place digitization programs for the last ten years, Operation Enduring Freedom (OEF) and Operation Iraqi Freedom (OIF) were the first major operations that pointed the way toward how the Army will communicate in the future. We begin by briefly discussing some implications on Army needs for bandwidth, including how Army communications have evolved toward much greater dependence on digital networks supporting mobile on-the-move tactical forces.

Recent Operations

The Army, and more generally the Department of Defense, has greatly increased communications capacity and utilization since September 11, 2001. This increase was evident in the recent OEF and OIF operations. Table 1 summarizes this growth.

Table 1
Growth in Communication Infrastructure and Demands Between September 11, 2001 and April 3, 2003

Communications System	Capabilities Growth
Defense Red Switch Network (DRSN)	Infrastructure increased 400%
Secure Internet Protocol Router Network (SIPRNET)	Capacity increased 292%
Non-Secure Internet Protocol Router Network (NIPRNET)	Capacity increased 509%
Defense Switched Network (DSN)	Infrastructure increased 138%
DISN Video Secure Global (DVSG)	Usage increased 1150%
Enhanced Mobile Satellite Services (EMSS)	Users increased 300% and usage increased 3300%

SOURCE: Raduege (2003).

The Defense Red Switch Network (DRSN) provides secure communications between fixed facilities. The Secure Internet Protocol Router Network (SIPRNET) provides a secure (up to Secret) digital internet network supporting messaging, database sharing, and collaboration tools. The Non-Secure Internet Protocol Router Network (NIPRNET) provides digital network access at an unclassified level with access limited to authorized users. The Defense Switched Network (DSN) provides unclassified communications to DoD users at fixed facilities. The DISN (Defense Information System Network) Video Secure Global (DVSG) provides classified videoteleconferencing to DoD users in fixed locations. The Enhanced Mobile Satellite Services (EMSS) is based on the Iridium satellite communications system to provide global voice and limited data access to satellite communications.

All of these communications systems have more than doubled their physical size since 9/11. In addition, usage of video communications and connections to mobile users (DVSG and EMSS) has increased by more than an order of magnitude in the same time period.

Based on Table 1 and other lessons learned from OEF and OIF (USMC, 2003; Wallace, 2003; Moran, 2003; 3ID, 2003; Shaaber, Hedberg, and Wesson, 2003; Raduege, 2004), some implications on future communications usage can be identified.

- Globally connected communication is desired for mobile tactical forces while they are moving in terrain ranging from highly mountainous to relatively flat.
- Ground forces will operate with increased dispersion (twice the maneuver area with half the forces compared to Desert Storm), straining and many times exceeding the communications capability of terrestrially based communications.
- Users at all echelons will demand access to communications networks for services and information, and that access will include all classification levels.
- Commanders will rely on blue force tracking to know where their forces are located for both planning and execution of missions.

- Real-time video and access to videoteleconferences will be demanded at all classification levels.

Unique Problems for Mobile Forces

Army after action reports cite the unique problems of ground forces in communicating large amounts of data to moving forces (3ID, 2003; Shaaber, Hedberg, and Wesson, 2003). While moving, forces are not able to connect using landlines or high-capacity satellite terminals. Instead they must rely on antennas that are able to transmit and receive over a large range of directions to account for the movement of antennas on vehicles moving over possibly rough terrain. These antennas have little gain, or directionality, so that data rate throughput is limited to tens of kilobits per second.

This data rate is sufficient to convey compressed voice with reasonable quality, as specified in NATO Standardization Agreement (STANAG) 4591. This STANAG is based on quality-of-service tests of different voice coders, with acceptable quality at data rates as low as 2.4 kbps (NATO, 2002). Other user needs, however, are not supportable at this data rate. For instance, a single medium-quality small-area image consisting of 5MB of data would take over eight minutes to transmit. A high-quality teleconference using current technologies would require a data rate of 384 kbps.

The data rates needed to provide the services listed in Table 1 far exceed those that are available to mobile users. As a result, high-quality services such as imagery and videoteleconferences were only available to higher-echelon commanders (brigade and above) and only when they were able to stop and set up high-capacity satellite radio antennas. Lower tactical echelons were not able to access these services and relied on lower data rate communications links when moving.

Ongoing Efforts to Increase Throughput to Mobile Users

The Army Communications and Electronics Command (CECOM) is actively researching new communications networking technologies to drastically improve communications network performance (CECOM, 2002). The Multifunctional On-the-Move Secure Adaptive Integrated Communications (MOSAIC) Advanced Technology Demonstration is examining new technologies to improve quality of service, to allow prioritization of user communications, to increase security of communications, and to use compression to reduce the load on communications. The MOSAIC program and follow-on efforts are working to develop and improve communications for the future force.

The Army has also contracted to provide improved communications networking to static forces on a global basis (CBO, 2003). This technology allows users in the logistics community to dynamically manage network performance to allocate scarce communications capacity to meet urgent user needs. Less urgent needs are spread out in time to use networks when there is slack capacity available. This technology focuses on network management and is currently used on the existing global information grid.

At the same time, commercial industry is actively developing new technologies to connect mobile users of cellular phones to transmit and receive images and video. Current cellular phones are frequently equipped with low-resolution cameras that allow snapshots and video to be transmitted over cellular land-based and satellite networks. The video from these phones has low-resolution images with low frame rates, resulting in jerky motion. This type of technology has been used, however, by news organizations such as CNN to allow reporters to file stories in real time using minimal equipment. While the images are low quality in a technical sense, the timeliness of the reports and the proximity to ongoing events have proved effective.

DoD is also proposing new operational concepts based on networked forces, i.e., network-centric operations (GIG, 2002). This concept calls for military forces to be interconnected on a global basis to share information and develop and execute plans that are more comprehensive and timely than any adversary can match. DoD's plans include globally based Network-Centric Enterprise Services (NCES) and new communications systems to provide broadband access to static and global forces. These plans are for future force operations in the 2012-and-beyond timeframe.

Implications for the Army and Motivation of This Report

The convergence of evolving user needs and ongoing technological advances provides the Army an opportunity to rapidly field capabilities to its tactical forces to improve communications services.

The operational implications of OIF lead to a number of new challenges to Army communications:

1. Connectivity to tactical forces will be needed in an environment where forces are mobile and dispersed, in many cases, beyond the range of terrestrial communications links. This leads to systems that use satellites and/or airborne nodes to maintain connectivity.
2. Users will rely on communications networks for services and information.
3. Users will need a higher quality of service to ensure connectivity, secure access, and real-time audio and video. This will increasingly strain available communications capacity.

All of these lessons imply that the Army will need to make the best use of available bandwidth. This report discusses how existing data compression and network management techniques could be applied in the near to medium term to improve performance prior to the fruition of the Army's future force or the DoD's network-centric concepts.

It is important to note that previous studies (CBO, 2003; Hillman et al., 2002; JHU, 2002; Joe and Porche, 2004) have estimated potential shortfalls in bandwidth capacities in the future force of over an order of magnitude. These estimates already assume a degree of compression comparable to the estimates used in this report. The recommendations of this report are focused on improving near- to mid-term capabilities with existing techniques. These techniques will not "solve the bandwidth bottleneck" but would contribute to improving performance with minimal impact on existing networks.

Organization of This Report

The next four chapters discuss various aspects of compression, beginning with a general discussion of compression, followed by chapters on lossless compression and lossy compression of images and video. A chapter describing commercial network acceleration technologies follows, and the report concludes with a summary and a set of recommendations.

CHAPTER TWO

Overview of Data Compression

The objective of compression is to reduce the amount of data required to store or transmit digital information. Digital information can take many forms. Some common types of digital information are:

- *textual documents* such as email messages, memos, letters, and other documents;
- *images* such as drawings, illustrations, and photographs; and
- *temporal media*, or signals that change over time, such as audio and video.

For each of these types there are standard ways of storing or transmitting the information, which have been determined by convenience, simplicity, and tradition. Typically, the price of this convenience is a larger data size than is absolutely necessary to convey the information. Compression is the process of reducing the size of the data.

Storing and Transmitting Digital Data

All digital data is stored as a series of *bits* containing ones and zeros. A single bit could be used to encode two distinct messages. That is, a zero can be used for one possible message and a one can be used for another possible message. Two bits can encode up to four distinct messages, represented by 00, 01, 10, and 11. A series of eight bits is a *byte*, and a byte can encode 2^8, or 256 distinct messages. Information theory states a general relationship between the size of the encoding and the number of possible messages that it can encode: an encoding scheme that uses n bits can encode up to 2^n distinct messages. Typically, large messages such as text documents or images are first broken down into small units, and each unit is encoded separately. For example, a text document is typically encoded as a series of individual characters, while an image is typically encoded as a series of picture elements, or *pixels*, starting at the top left corner of the image and scanning from left to right, top to bottom. In both cases, each unit is typically encoded in a byte or a few bytes.

The same information can be digitally encoded in multiple ways, some of which are more compact, or efficient, than others. For example, consider Paul Revere's alleged message about whether the British were coming by land or sea. Revere chose to use a signal of one or two lanterns in a church tower for his message. This encoding scheme was a very efficient way of sending the small set of messages he wanted to be able to send. Virtually any other method of encoding his messages would have required more data. For example, if he had tried to spell out the text "The British are coming by sea" in lights, he would have needed many more than two lanterns. Revere's encoding had one bit (he had only one choice to make: either one lantern or two), so he could only encode 2^1, or two distinct messages.

By choosing this very efficient coding scheme, Revere limited the variety of messages he could send. In addition, every message recipient had to be notified in advance how to interpret the two signals. Once Revere set up this scheme, if he had wanted to send the message "The British are coming by air," there would not have been any way for him to do it. This limitation was acceptable to him because he had only two possible messages he would want to send. The less efficient scheme of spelling out a word with lights would have given him much more flexibility in exactly what messages he could send, at the expense of a much larger data size.

Sources of Data Compression

There are several potential methods of data reduction that can be exploited by compression algorithms.

The first is to improve coding efficiency. Coding inefficiency arises when the standard way of encoding a message unit uses more bits than necessary. For example, using a byte (8 bits) to encode 26 alphabetic characters is not efficient because a byte can actually encode up to 256 distinct characters. Actually, 5 bits would be sufficient to encode 26 alphabetic characters, because 5 bits can encode up to 32 distinct messages.

A second way to achieve data reduction is to reduce redundancy in messages. For example, suppose a security camera is monitoring an empty room by transmitting a video frame every 1/30th of a second, the frame rate for video in the NTSC (National Television System Committee) standard. As long as nothing moves in the room and the lighting does not change, frame after frame of the video from that camera will be identical (ignoring noise for the moment). Thus, transmitting every frame of the video when nothing is happening in the room is redundant. A compression scheme might take advantage of this redundancy by only transmitting frames when they differ from the previous frame, reducing the total amount of data transmitted.

A third method for achieving compression is to approximate the message rather than transmitting it exactly. For example, consider the same security camera watching the same room, but with an insect crawling across the wall. The movement of this insect will cause successive frames of the video to differ slightly. This would limit the ability to take advantage of message redundancy described above. However, changes as small as the motion of an insect might be viewed as insignificant for security purposes. A compression scheme might take advantage of this by suppressing very tiny changes in the video from one frame to the next. This would restore the ability to obtain compression through message redundancy. However, this type of compression comes at the cost of accuracy. The video received would no longer be identical to the original video captured by the camera, although it might be considered an acceptable approximation. This type of compression is called lossy compression. In lossy compression, some information is lost, such that the recipient of the message cannot *exactly* reconstruct the original data.

Improving Coding Efficiency

For illustration of these various compression methods, let us consider textual documents.

ASCII (American Standard Code for Information Interchange) is a standard way of digitally storing textual documents that use the Roman alphabet. In ASCII, each letter, number, space, or symbol occupies one byte. This size was chosen because computer memory is organized in multiples of bytes, making the storage and retrieval of any particular section of a textual document very easy. Each byte in memory corresponds to exactly one character in the text, independent of context.

A byte can hold 256 different values, from 0 to 255. ASCII assigns each letter, number, and symbol a standard and unique value. For example, the lowercase letter “a” is given the decimal value 97, which is the same as the binary value 01100001; the numeral “1” is given the decimal value 49, which is the same as the binary value 00110001. These assignments of numeric values for each symbol are somewhat arbitrary, but all of them are in the range of 0 to 255 and all of them occupy exactly 8 bits. ASCII distinguishes between uppercase and lowercase letters and includes many symbols, as well as some unprintable characters called control characters.

Many messages do not make use of the full ASCII character set. For example, consider this telegram: HI MOM STOP SENT 100 DOLLARS STOP WILL CALL NEXT WEEK STOP JUNIOR. In ASCII, this 65-character message occupies 65 bytes. In messages like this example, where only uppercase alphabetic characters, numbers, and spaces are used, there are only 37 possible character values, which does not fully utilize the range of 256 values provided by ASCII. This set of 37 different values could be encoded in 6 bits rather than the 8 bits used by ASCII. Six bits allows for up to 64 distinct values. So we could create a new, more efficient coding scheme for telegrams that only used 6 bits. This more efficient coding scheme would enable transmission of the 65-character telegram in about 49 bytes instead of 65 bytes.

Of course, with this new encoding scheme, it would not be possible to send telegrams with lowercase characters or punctuation marks. If the encoding scheme were expanded to include uppercase and lowercase characters, and all of the other symbols on a typical computer keyboard, 7 bits (128 values) might still be sufficient. Thus, for most ordinary text documents ASCII is not the most efficient way of encoding the message because it uses 8 bits for every character when 7 bits would suffice. If we were to look at the binary values for the above telegram encoded in ASCII, every byte would begin with a 0 in the most significant bit. Since this bit is always zero, it has no information value. We could choose a different encoding scheme that eliminated this zero and only used 7 bits, and the entire message could still be represented intact. This would be a 12.5 percent reduction in the amount of storage required for the message.

The inefficiency of ASCII is usually tolerated because of the convenience and context-independence of using individual bytes for storage. In a 7-bit encoding scheme, any particular byte (8 bits) in the computer's memory will hold fragments of two characters (which are 7 bits each). Furthermore, contextual information is now necessary to correctly extract characters because the pattern of how characters are arranged varies across the message. The first byte in the message holds seven bits from the first character and one bit from the second character; the second byte holds six bits from the second character and two bits from the third character; the third byte holds five bits from the third character and three bits from the fourth character; and so on. So, when looking at any particular byte, it is necessary to know that byte's position in the overall message to calculate how to properly extract the two fragments of the characters represented by that byte. ASCII's one-to-one mapping of characters to bytes, and the context independence of extracting characters from the bytes, has been lost in this 7-bit encoding scheme, thus adding to the complexity of encoding and decoding messages.

Reducing Message Redundancy

To illustrate the second potential source of compression, message redundancy, let us consider the same telegram: HI MOM STOP SENT 100 DOLLARS STOP WILL CALL NEXT WEEK STOP JUNIOR. Recall that if this message were transmitted in ASCII, it would occupy 65 bytes because the message is 65 characters long, including spaces.

This message has a 6-character pattern that is repeated in three places—" STOP "—the word *stop* surrounded by a space on either side. In total, 18 of the 65 characters in the message are used by this pattern. Suppose we replace this pattern with a single symbol—let's use a percent sign—so the message reads: HI MOM%SENT 100 DOLLARS%WILL CALL NEXT WEEK%JUNIOR. The new message contains only 50 characters, a 23 percent reduction in the message's size.

When the compressed message is received, the original message can be restored exactly by replacing all of the percent signs with the " STOP " text pattern. To do this, however, the receiver of the message must know that this encoding is being used. If this were not agreed upon in advance, a *dictionary* would have to be transmitted along with the message to enable the receiver to decode the compressed message. Transmitting such a dictionary, of course, requires sending additional data, reducing the total amount of compression that is achieved.

This substitution works because there were no other percent signs in the original message. If the original message had said SENT 100% instead of SENT 100 DOLLARS, percent signs could not be used for encoding the " STOP " pattern. Otherwise, the decompression algorithm would produce an incorrect decoding of SENT 100%, decoding it as SENT 100 STOP instead.

Another thing to note about this substitution is that it is sensitive to the kinds of messages being transmitted. The substitution of percent signs for " STOP " might work well for telegrams, but it would be unlikely to work well for transmitting other kinds of text documents, or other kinds of data such as images or video. However, in these other types of data files, other patterns of redundancy may emerge. For example, in long English text documents the pattern " the " might be very frequent and a potential source of compression. But this pattern might not appear very frequently in Spanish text documents. Similarly, large blocks of solid color, such as blue sky, might appear in photographic images.

Probability Models

In order to take advantage of the redundancy in a message, it is necessary to know what the high-frequency patterns are likely to be. This is called a *probability model* for the message. A probability model for English would assign high probabilities to patterns like " the " and low probabilities to uncommon words or patterns. Then a compression algorithm can concentrate on these high-frequency patterns for potential sources of compression. Documents that are a good match for the probability model will be compressed better than documents that do not match as well. For example, using a probability model for English text documents for compressing an image would be unlikely to work very well.

Some compression algorithms rely on probability models that are computed in advance. For example, if many textual documents are to be transmitted, a corpus of typical documents could be analyzed in advance to form the probability model. Similarly, if many aerial images will be transmitted, a corpus of such images could be used to compute a probability model for them. In either case, both the compressor and the decompressor would use the model for all messages that are transmitted.

Other compression algorithms incrementally build a probability model as they compress each message. These algorithms are *adaptive*. If they are fed English text documents they will form a different probability model than if they are fed Spanish text documents or images. These algorithms might assign a high probability to " the " in a typical English document, and high probabilities to " el " and " la " in a Spanish document. However, a disadvantage to these algorithms is that the probability model changes for every document, and the decompression algorithm needs the model to decode the message. When using this technique, the compressor and decompressor cannot agree in advance to use on a single probability model, so the compressor must somehow transmit the probability model to the decompressor for use in decompression. Clever algorithms are able to transmit the probability model to the decompressor with no additional overhead beyond the transmission of the compressed message itself. The decompressor then incrementally builds the probability model as it is decoding the message data.

Message Approximation (Lossy Compression)

In lossy compression, some inaccuracies are tolerated in the decompressed message in order to improve compressibility. For example, considering the same telegram message, HI MOM STOP SENT 100 DOLLARS STOP WILL CALL NEXT WEEK STOP JUNIOR, suppose we decided to eliminate the " STOP " patterns completely: HI MOM SENT 100 DOLLARS WILL CALL NEXT WEEK JUNIOR. Some information in this compressed message has been thrown away, and the recipient cannot precisely reconstruct the original message. In this example, the number of STOPs in the original message, and their exact placement, have been lost. This loss of information might be tolerated if the important information is still conveyed. For example, if Junior's mother received this message, she would probably still understand it.

However, with this lossy encoding scheme there are other messages that would be encoded the exact same way. One example is: HI STOP MOM SENT 100 DOLLARS STOP WILL CALL NEXT WEEK STOP JUNIOR. In the original telegram, it was implied that Junior sent the 100 dollars, while this second message indicates that the mother sent the 100 dollars. Using the lossy compression algorithm where STOPs are discarded would result in these two different messages being indistinguishable by the recipient. This illustrates a risk of lossy compression: that important information will be lost, leading to a garbled or erroneous message.

As with the use of probability models for eliminating message redundancy, lossy compression strategies vary for different types of data. In English text, eliminating the patterns " a " and " the " might be a useful strategy because these words do not convey much important information. In images, the type of information that is typically discarded is the information that is less useful to the human visual perception system.

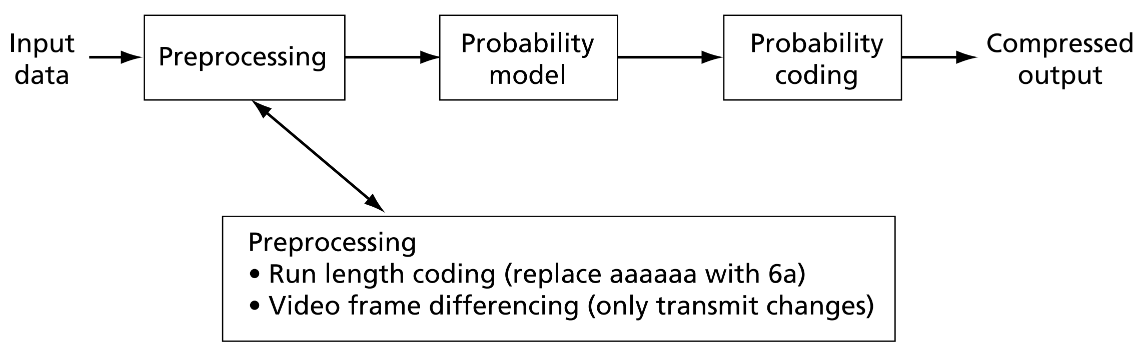
The following chapters describe popular lossy and lossless algorithms in more detail. We begin in Chapter Three with a discussion of lossless compression. Lossless compression results in greater communications throughput without any loss of information. The chapter describes how this is achieved and the potential quantitative benefits. Subsequent chapters describe lossy techniques, where information is intentionally discarded to further improve compression.

CHAPTER THREE

Lossless Compression

The objective of lossless compression is to produce compressed output that is smaller than the input data but to do so in a reversible manner, so that the input data can be *exactly* reconstructed from the compressed data. The steps in lossless compression are preprocessing, probability model, and probability coding. The method used for reconstructing the original data, or decompression, is the inverse of the process in this diagram. Figure 1 shows the steps of lossless compression, with some details of the preprocessing step.

Figure 1
Lossless Compression Step 1: Preprocessing



RAND *TR216A-1*

Lossless Compression Step 1: Preprocessing

In some compression algorithms, the input data goes through a transformation step prior to the compression process. This transformation step is called preprocessing, and it takes advantage of contextual information to improve the compressibility of the data.

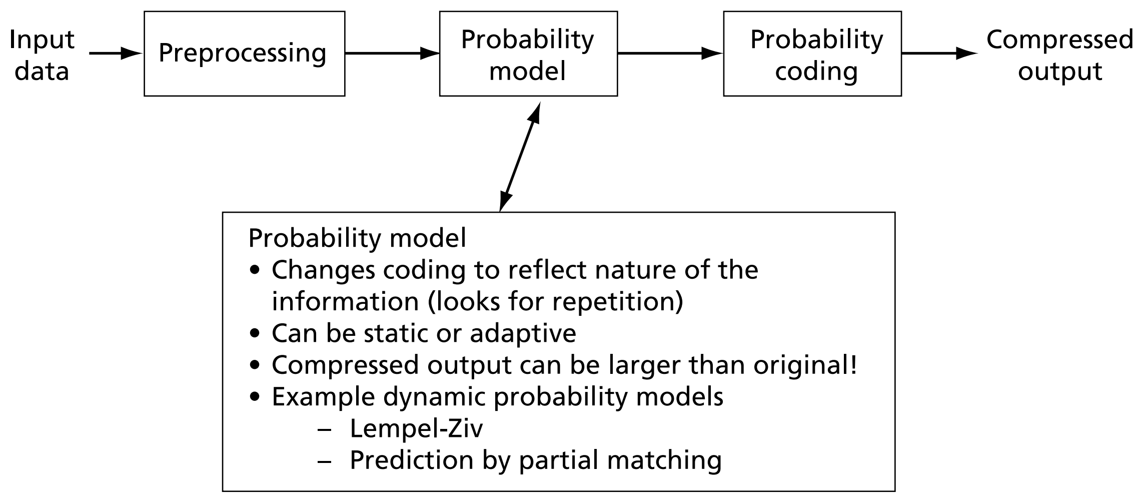
An example of preprocessing is run-length encoding. Run-length encoding takes sequences of repeated patterns and replaces them with a (count, pattern) pair. For example, a run-length encoder might replace the text pattern aaaaaabbbbb with (6,a)(5,b). Another example of run-length encoding is used by fax machines, where sections of solid white between lines of text or at the end of a document can be compressed very efficiently. A fax machine might say the equivalent of "the next six inches on this page are blank" instead of transmitting this section of the page as if there were ink patterns on it.

Another kind of preprocessing that might be used by a lossless compressor for video streams is called frame differencing. In frame differencing, a prior video frame is subtracted from the current frame. To the extent that the two frames are similar, this will cause large portions of the resulting frame to be zero. Run length encoding could then be effectively applied to these sections. The decompressor can still exactly reconstruct the current frame, by starting with the prior frame and adding the decoded data for the current frame to it.

Lossless Compression Step 2: Probability Model

Figure 2 expands on the next step in lossless compression, the use of a probability model to capture information.

Figure 2
Lossless Compression Step 2: Probability Model



RAND *TR216A-2*

The central component of a lossless compression algorithm is a probability model for the input data. A probability coder consults this model to determine how it will encode input data patterns in the compressed output. Input patterns that have high probabilities are coded with short output patterns, while input patterns that have low probabilities are coded with longer output patterns. If the input data is consistent with the probability model, there will be a preponderance of high-probability input patterns, and thus the coder will make frequent use of the short output patterns. The overall result would be a compressed output that is smaller than the input data. However, a law of compression states that there also exists some input data that is not a good fit for the probability model, and for that input data the compressed output will actually be larger than the input.

Static Versus Dynamic Probability Models

Some probability models are static and do not vary for different input data. An example of a static probability model would be the frequencies of word occurrences in English text documents. Such a probability model might work quite well for compressing various kinds of text such as books, emails, and newspaper articles, but it would probably do poorly for images and other binary input data. Because no single static probability model would work well for all kinds of input text, many of the most effective compression algorithms have dynamic probability models. These algorithms build or modify the probabilities as input data is encountered.

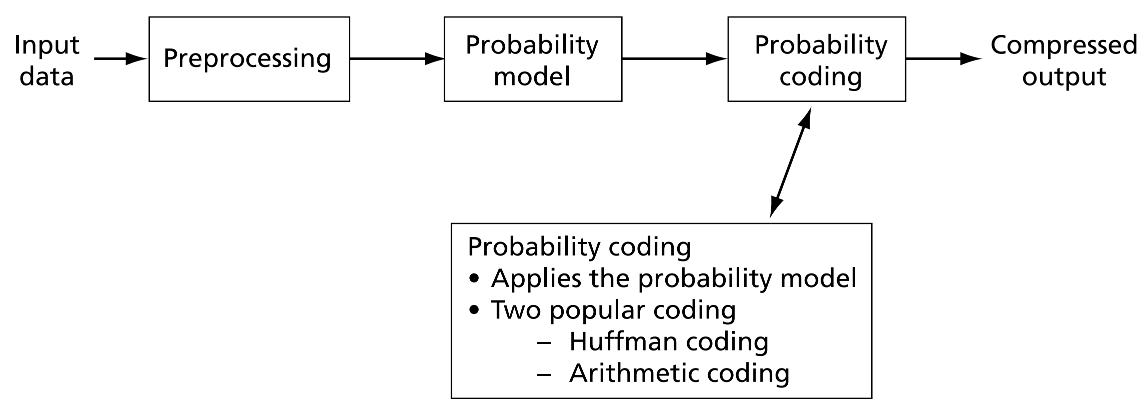
Examples of dynamic probability models are Prediction by Partial Matching, and a family of dictionary-based algorithms based on a family of compression algorithms named Lempel-Ziv (Ziv and Lempel, 1977). The Lempel-Ziv family includes compression standards such as zip, gzip, compress, v.42bis (used by modems), and gif (used for encoding images).

These algorithms incrementally build a dictionary of the patterns encountered in the input data and track the frequencies of occurrence of the patterns. The advantage of this method is that the probability model does not incorporate any *a priori* assumptions about the characteristics of the input data. Therefore, these algorithms are good for multipurpose use: no matter what the input data is, the algorithm can create a suitable probability model and therefore achieve a reduction in data size. However, an important challenge with dictionary-based probability models is that the dictionary must somehow be sent to the receiver along with the compressed data. Otherwise, the decompression algorithm would not know what patterns to use in reconstructing the original data. If the size of the compressed data plus the size of the dictionary is not smaller than the input data, the compression would not be a success. A key feature of the Lempel-Ziv family of algorithms is that the decompression algorithm rebuilds the dictionary incrementally as it works through the compressed data. That is, no additional data must be sent describing the dictionary itself.

Lossless Compression Step 3: Probability Coding

The third step in lossless compression involves the probability coding of the data based on the probability model, as shown in Figure 3.

Figure 3
Lossless Compression Step 3: Probability Coding



RAND *TR216A-3*

Given a probability model, each element in the input data is then looked up in the model and coded accordingly. As mentioned earlier, high-probability patterns are assigned small output patterns, and low-probability patterns are assigned larger output patterns.

Almost all lossless compression algorithms use one of two popular probability coding schemes: Huffman Coding or Arithmetic Coding. Huffman Coding is optimal when all of the probabilities happen to be powers of two—something that would rarely occur in real-world applications. In practical use, Arithmetic Coding has a slight data-reduction advantage over Huffman Coding, but it is more computationally expensive and is subject to patent royalties.

Performance of Lossless Compression

Lossless compression algorithms tend to work well for textual data. They can also be used on image data, but much higher compression ratios can be achieved by using lossy compression algorithms.

Table 2 shows the range of compression that can be achieved using two technologies employing different probability models. The range of performance depends on the nature of information to be compressed, i.e., how many repetitive patterns are embedded.

Table 2
Compression Ratios Achieved by Lossless Algorithms

	Lossless Compression	
Technology	**Example Standard**	**Compression Ratio**
Lempel-Ziv	gzip	1:1 to 9:1
Predictive/Huffman Coding	Lossless JPEG	1.3:1 to 3:1

The next two chapters discuss two lossy compression algorithms, Joint Photographic Experts Group (JPEG) and Moving Picture Experts Group (MPEG). These lossy compression schemes build upon lossless compression: lossless compression is one component of the overall lossy algorithms.

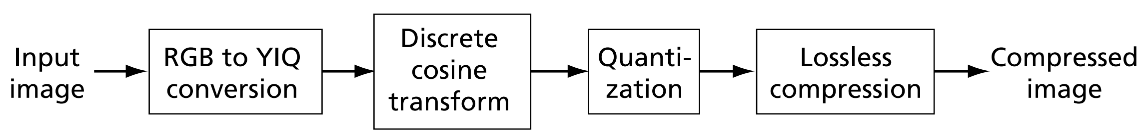
CHAPTER FOUR

Lossy Compression of Images

As stated above, when lossy compression is used, the input image cannot be *exactly* reproduced upon decompression. The key to successfully utilizing a lossy algorithm is that the resulting approximation of the original image must be close enough to original to be acceptable. The amount of divergence that is acceptable will vary across applications. For example, an image that will be enlarged for a museum exhibit may not be able to tolerate much inaccuracy, while an image from a security camera might tolerate much more inaccuracy. In general, lossy compression algorithms can be tuned to vary the amount of compression, and the corresponding amount of lossiness.

A very popular lossy compression algorithm is JPEG, which is used for image data. Figure 4 shows the major steps in the JPEG compression process.

Figure 4
Model of JPEG Compression



RAND *TR216A-4*

First, color images are converted into a format that will facilitate compression, called YIQ.[1] Then, a discrete cosine transform (DCT) algorithm is applied. After this, the data is quantized, discarding some of the less important information. Finally, a lossless compression algorithm further compresses the data. As in all types of compression, these steps are reversed to decompress the image.

For the remainder of this chapter we will describe in more detail these steps of JPEG compression.

Image Representation

First, recall how images are represented digitally. Images are almost always stored as rectangular objects. The rectangle is divided into square dots called *pixels* (a contraction of "picture elements"). An uncompressed image is stored in a computer file as a sequence of pixel values,

[1] The components of YIQ are luminance (Y), in-phase (I), and quadrature (Q).

starting with the pixel value at the top left corner of the rectangle and moving across a row of pixels to the right, followed by the next row, and so forth until all of the pixels are stored. This storage format is called *scanline* format.

For monochrome (or grayscale) images, each pixel is represented by an 8-bit value ranging from 0 to 255, where a higher number represents a brighter pixel and a lower number represents a darker pixel. White is 255 and black is 0, while a dim gray might be 64. When the input image is a color image, it is usually represented as a set of three monochrome images representing color planes: red (r), green (g), and blue (b), and each of these is 8 bits in depth, for a total of 24 bits. This is known as RGB (red, green, blue) format. For each pixel in an RGB color image, the color and intensity are jointly determined by adding the red, green, and blue components for that pixel, using an additive color model. In this model, white is represented by r=255, g=255, and b=255, where all components have the maximum value; black is represented by r=0, g=0, and b=0. A bright yellow would be r=255, g=255, b=0, because adding a bright red to a bright green yields a bright yellow; a dim gray would be r=64, g=64, b=64, because adding the three colors together with equal intensity yields a monochrome pixel.

As an example, a typical image from a low-end digital camera of 1.3 megapixels might produce a rectangle that is 1280 pixels wide and 960 pixels tall. This is large enough to fill most of a typical computer screen. The total number of pixels in this image is 1280 times 960, or 1,228,800 pixels. Since each pixel occupies three bytes (24 bits), the total number of bytes occupied by this image in uncompressed form is 3,686,400, or approximately 3.5 megabytes.

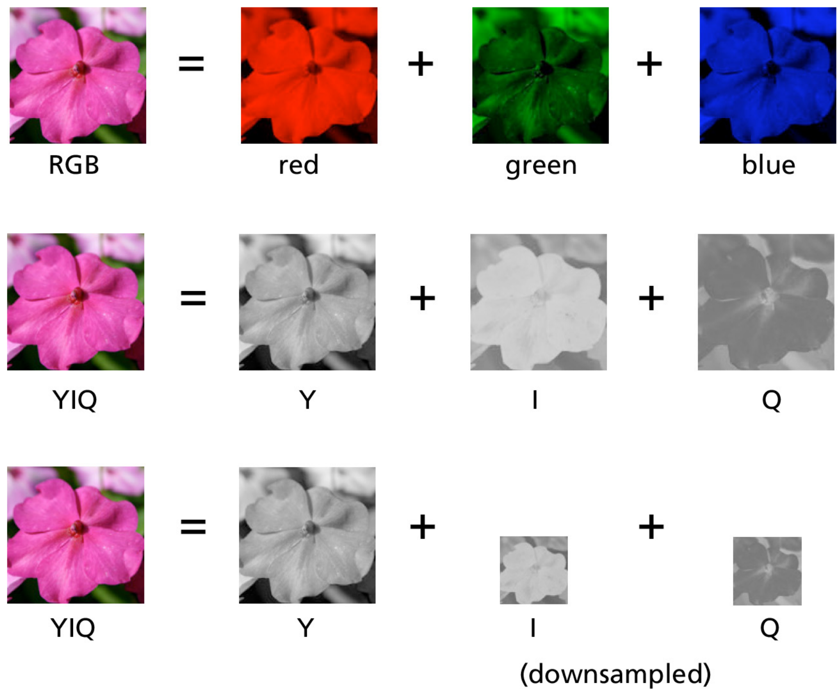
When the computer is displaying an image, it uses the first three bytes to determine the color of the pixel in the top left corner, the next three bytes to determine the color of the pixel just to the right of that one, and so on across the top row of 1280 pixels. Then it moves to the next row of 1280 pixels, and the next, until it has completed the 960th row. Because each individual pixel is very small and they are so close together, we do not usually notice the individual pixels. Instead, our visual system interprets the entire assemblage of pixels as a picture.

JPEG Compression Step 1: Color Conversion and Downsampling

Figure 5 illustrates the first step in JPEG compression, which is color conversion and downsampling. The top line of the figure shows a color image broken down into its red, green, and blue components. Each of the components is a monochrome image. Monochrome images are usually displayed using grayscale, but in this illustration we are using monochromatic color images for illustrative purposes.

JPEG works with either color or monochrome (grayscale) images. With color images, the first step in the JPEG process is to convert the red, green, and blue pixel values into a different representation called YIQ. This is a standard representation used by broadcast television. The second line of the illustration shows an image broken down into the YIQ components. As with RGB, the YIQ breakdown of three monochrome components does not discard any information.

Figure 5
Color Conversion and Downsampling



RAND *TR216A-5*

YIQ encoding puts all of the intensity information into a single luminance image (Y). The other two components of YIQ contain the color information, often called chroma. In Figure 5, the two chroma components are simulated. One reason television broadcast uses this standard is that it affords a way to provide backward compatibility with black and white televisions. The Y component is exactly the information needed by black and white televisions; the chroma components were added to TV signals in a way that would not interfere with a black and white television's ability to receive this Y signal.

If JPEG is already starting with a monochrome image, the YIQ conversion is not necessary. There is no color information in the image, so the I and Q components would be empty. A monochrome image is equivalent to the Y component of a YIQ image.

While the conversion from RGB to YIQ itself does not compress the image or lose information, most JPEG algorithms do "chroma downsampling," which reduces the spatial resolution of the I and Q components by one-half in each dimension. This is done by working with 2×2 squares of pixels. That is, each set of four pixels (arranged in a square) is replaced by a single average I value and a single average Q value rather than four of each. This reduction in color resolution is acceptable because the human visual system does not typically notice when color information is approximated in this way. Our visual system is much more sensitive to luminance than chroma.

For example, in chroma downsampling the first two pixels from the first row of the image and the first two pixels from the second row of the image are considered together. These four pixels originally have 3 bytes each of Y, I, and Q information, for a total of 12 bytes. After chroma downsampling, each pixel still has its Y byte, but they now share a single set of average chroma values: a single byte of I and a single byte of Q. These I and Q values are simply the averages of the four individual I and Q values for the set of pixels. The resulting data for these four pixels totals to 6 bytes: 4 bytes of Y, one byte of I, and one byte of Q.

Although the reduced color data does not significantly reduce the fidelity of the image, chroma downsampling has achieved a data reduction of 50 percent. The cost is some loss of data, and the original image can no longer be exactly reconstructed from the compressed image. This is the first of several lossy steps in the JPEG algorithm. Some JPEG compressors offer the option to turn off this chroma downsampling step.

Each of the Y, I, and Q components are monochrome rectangular sets of pixels. In the case of our 1280×960 digital camera image, we now have a 1280×960 Y image, and 640×480 I and Q images. For the remainder of the compression steps, JPEG treats each of these separately using the same set of steps. Therefore, the following descriptions will detail the operations performed on a single one of these three images.

JPEG Compression Step 2: Discrete Cosine Transformation

The DCT does not work on an entire image at one time. Instead, for the next step in the process, the DCT algorithm breaks the image into *blocks*, which are square pieces of the image measuring 8 pixels by 8 pixels.

The DCT is applied to each block individually. DCT relies on Fourier Series theory (Carslaw, 1952), a trigonometric principle that any function can be represented by a series of sine (or cosine) functions of various frequencies and amplitudes. In JPEG, a set of 64 cosine basis functions is predefined for use in the series. The basis functions vary in frequency in the horizontal and vertical dimensions. A set of 64 basis functions is shown in Figure 6. Each square in the figure is one of the basis functions and consists of an 8×8 set of pixel values.

For a given 8×8 block from the input image, the DCT algorithm calculates a coefficient for each of the 64 basis functions. Each coefficient represents the amplitude at which its basis function is added into the series. The resulting 8×8 block of coefficients thus represents the linear superposition of basis functions that best describes the pixel block. If a basis function is not needed for a particular series, the coefficient is zero.

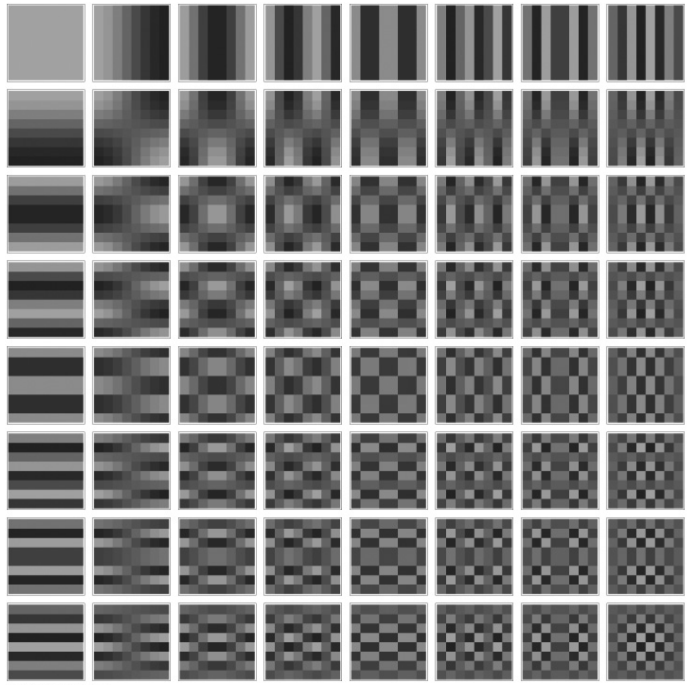
The input block contains 64 bytes (8 pixels by 8 pixels, one byte each). The result of the DCT transformation is a new set of 64 8-bit values, the coefficients of the basis functions. In principle, the DCT does not introduce loss, but because the calculations are based on transcendental functions,[2] the computer cannot calculate them with perfect accuracy. Therefore, this step is not exactly invertible and results in loss of data, although the loss is minor compared with the other lossy steps in the JPEG algorithm.

JPEG Compression Step 3: Quantization

The next step in the JPEG compression process is quantization. Quantization is essentially a rounding operation, where some of the less significant bits are discarded.

[2] Transcendental functions are those that cannot be expressed in terms of algebraic formulas. Digital computing devices use approximations in computing these functions. Volder (1959) introduced a common algorithm for this approximation.

Figure 6
Discrete Cosine Transformation



RAND *TR216A-6*

The DCT coefficients can be arranged in order from the lowest frequency (the direct current (dc) component, which is the average intensity of the 8×8 block) to the highest frequency. Quantization is applied to each coefficient, but not uniformly. More bits are retained for the lower-frequency components than the higher-frequency components. Again, this choice is made for perceptual reasons: human vision is more sensitive to the lower-frequency components. Therefore, a lookup table of quantization values is used to decide how much of each coefficient should be discarded. This quantization table is transmitted along with the compressed image, and can vary from one run of the JPEG algorithm to the next. Most JPEG programs permit the user to select a "quality" value. This quality value selects a quantization table that discards either more or less of the resolution of the DCT coefficients. Each JPEG implementation can choose the exact values used in the quantization table because the decompression algorithm receives the table along with the image and will use it to reverse this step.

JPEG algorithms typically use different quantization tables for the luminance and chroma components. To optimize the JPEG algorithm for human viewing, psychovisual experiments can be performed to try to determine optimal quantization values for each of the Y, I, and Q components, in order to minimize the introduction of visible defects. In practice, however, this cannot be optimized for all compression situations because it depends on image characteristics, display characteristics, and viewing distance. Furthermore, the optimal settings for human viewing may not be optimal when the decompressed image must be processed or interpreted by machines. Overall, the quantization step is the primary source of lossiness in JPEG.

JPEG Compression Step 4: Lossless Compression of the New Image

The final step in JPEG compression is lossless compression of the quantized DCT coefficients.

After the quantization step, 8×8 pixel blocks may have zeros for the coefficients of many of the higher-frequency DCT components. In fact, for regions of the image that have solid colors, all of the components except the dc component might be zero. Furthermore, from one 8×8 block to the next, the dc component is likely to be similar. Therefore, for further compression, the dc component is separated from the alternating current (ac) components.

Each dc coefficient is coded as the difference from the dc coefficient in the previous block. When adjacent blocks are similar, these difference values will be small numbers, increasing the compressibility of the series of dc coefficients. The ac coefficients, which may have long series of zeros, are especially amenable to run-length encoding compression.

The final step in JPEG takes advantage of the compressibility of these dc and ac coefficients by passing them through lossless compression algorithms. In general, the amount of compression achieved by this lossless compression step is dependent on the amount of quantization done in the prior step. Higher quantization introduces greater lossiness but increases the uniformity of the coefficients, yielding higher compression in this step.

Performance of Lossy Compression Algorithms

In practical use, the JPEG algorithm offers compression ratios of up to 100:1 (Balogh et al., 2000). However, when it is desirable to minimize visible defects and produce a decompressed image that is visually nearly indistinguishable from the original, the compression ratio achieved is usually much less, in the range of 12:1 to 16:1. When defects are visible, they usually reveal the 8×8 block pattern that is used by the algorithm. Since each block is independently computed, there are often discontinuities in color or intensity at the transitions between blocks. Table 3 illustrates the range of performance that can be achieved with lossy compression algorithms in comparison with lossless algorithms.

It is important to note that measures of quality are subjective. There are no accepted industrywide, universally applicable standards for measuring the amount of information lost in a lossy compression algorithm. Therefore, any application of lossy compression should be coupled with a metric for the amount of loss that is appropriate for how the decompressed information will be used (Hillman et al., 2002; JHU, 2002; Kosheleva, Mendoza, and Cabrera, 1999; Lai, Li, and Kuo, 1996; Rountree, Webb, and Marcellin, 2002; Yeh and Venbrux, 2002).

JPEG 2000 and Discrete Wavelet Transform

The new JPEG 2000 compression scheme is very similar to the model described above for JPEG (JPEG, 2003). It has many improvements over JPEG, such as a more sophisticated quantization step, but one major difference is the replacement of the discrete cosine transform (DCT) with a discrete wavelet transform (DWT). The DWT uses the same principle of representing image blocks as a sum of basis functions, but instead of using transcendental functions, the DWT offers a choice between a few sets of other basis functions. One of these sets of basis functions allows the coefficients to be calculated using integer arithmetic, eliminating rounding errors at this step, and thus providing a lossless compression option if the chroma downsampling and quantization steps are skipped. JPEG 2000 compression ratios can be as much as a factor of three greater than JPEG; however, this performance superiority is much smaller for high-quality imaging applications, where the advantage is about 10–20 percent (Christopoulos, Skodras, and Ebrahimi, 2000). At similar compression ratios, JPEG 2000 produces better image quality than JPEG.

Table 3
Compression Performance of Lossy and Lossless Algorithms

	Compression of Still Images			
	Lossless		Lossy	
Technology	Example Standard	Compression Ratio	Example Standard	Compression Ratio
Lempel-Ziv	gzip	1:1 to 9:1		
Predictive/Huffman Coding	Lossless JPEG	1.3:1 to 3:1		
Discrete cosine transform (DCT)			JPEG	3:1 to 100:1
Discrete wavelet transform (DWT)	JPEG 2000	1.5:1 to 4:1	JPEG	10:1 to 350:1

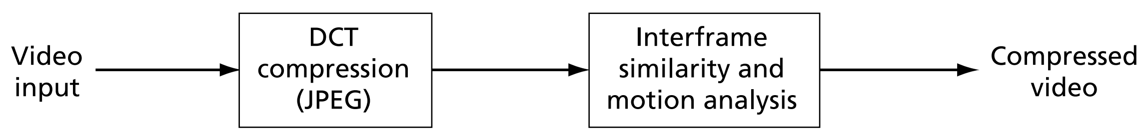
CHAPTER FIVE

Lossy Compression of Video

Video is simply a sequence of still images. In digitized NTSC video, there are 30 frames per second, and each frame is a 640×480-pixel 24-bit color image. One way to compress video would be to simply compress each individual frame using the JPEG algorithm. However, there are additional attributes of video that can lead to further compression. In particular, adjacent video frames are usually quite similar to each other. Since it would treat each frame individually, JPEG alone cannot take advantage of these interframe similarities.

MPEG builds upon JPEG to gain further compression from interframe similarities in video (MPEG, 2003). A model of the MPEG family of algorithms, which include MPEG-2 and MPEG-4, is shown in Figure 7.

Figure 7
MPEG Video Compression



RAND *TR216A-7*

The first step in MPEG is to process each individual frame with a discrete cosine transform compressor similar to what is used in JPEG. Then, additional analysis is performed to take advantage of interframe similarities.

MPEG Compression Step 1: DCT Conversion of Individual Frames

The first step in MPEG is to apply a DCT (JPEG) compression algorithm to each individual frame of the video stream. In actuality, the two steps in the MPEG process are more integrated than depicted in this diagram. The algorithms used in step 2 make use of intermediate results from the DCT compression process. However, for conceptual simplicity it is useful to portray the DCT step and the other analysis as distinct steps. For details of what happens in the DCT step, please refer to the description of the JPEG algorithm in the previous chapter.

MPEG Compression Step 2: Interframe Similarity and Motion Analysis

The essence of step 2 in the MPEG compression process is to take advantage of similarities among nearby frames in the video stream. Conceptually, this is done by occasionally transmitting independent frames, which can be wholly reconstructed without referring to any prior or subsequent frames, and between these independent frames transmitting information that allows additional frames to be derived from the independent frames.

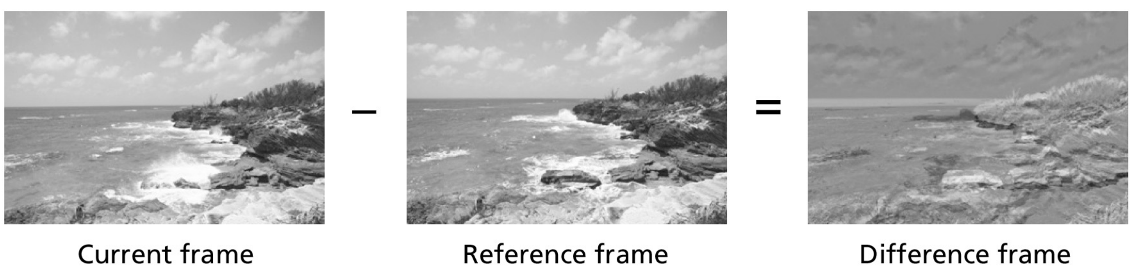
Interframe Similarity

The *independent* frames in MPEG are called I-frames. For each I-frame, MPEG passes the JPEG-compressed video frame straight through to the compressed video stream. Clearly, JPEG-compressed images can be decompressed without referring to any other frames in the video stream.

In addition to I-frames, JPEG computes dependent frames that cannot be reconstructed without referring to other frames in the video streams. These are P-frames and B-frames. P-frames are called *predicted* frames, and are dependent on the most recent I-frame or P-frame. B-frames are called *bi-directional* frames, and are dependent not only on the most recent I-frame or P-frame, but also on the next P-frame or I-frame. The prior and subsequent frames that are consulted to compress and decompress P- and B-frames are called *reference* frames.

To compute a P-frame, the reference frame, which is the immediately prior frame of video, is subtracted from the current video frame. This process is illustrated in Figure 8.

Figure 8
Frame Differencing with Interframe Similarity



RAND *TR216A-8*

Note that during decompression, the decompressed reference frame will be used to rebuild the P-frame. Since the reference frame will have been compressed with JPEG, the decompressed reference frame will not be identical to the original reference frame. For this reason, the reference frame used in the P-frame calculations is not the original reference frame from the input video, but a version of the reference frame that has undergone compression and decompression. This extra decompression step inside the compression algorithm adds to the computational demands of the algorithm.

Consider a video sequence where there is no motion. In this situation, each video frame would be identical, and the difference will be zero, allowing the P-frame to achieve a huge compression factor.

At some points such as scene changes or objects entering or leaving the scene, a better match for the P-frame might be a subsequent frame. One problem with using a prior frame as the sole reference frame in the video image is that it does not take advantage of the P-frame's similarity to subsequent frames. This is where B-frames come in. B-frames consider bi-directional similarities relative to two reference frames: the closest prior and subsequent P- or I-frames. This additional information allows B-frames to be compressed even more than P-frames, at the cost of additional computational complexity. However, the most computationally expensive part of the P-frame calculation is motion analysis.

Motion Analysis

In video there is almost always motion, so the computation of the predicted frames also includes analysis to account for this motion. This analysis calculates estimates of how various blocks in the reference image have moved from the reference frame to the predicted frame. Based on these motion estimates, blocks from the reference frame are shifted before the subtraction is done. A similar process is used for both P-frames and B-frames. For B-frames, both the prior I-frame and the following P-frame are used in computing the motion analysis.

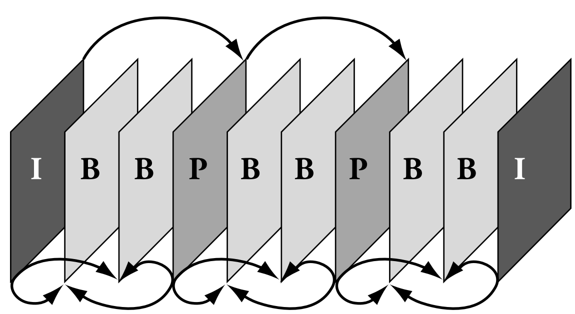
Usually the motion-adjusted reference frame will be a better match to the P-frame or B-frame, and the subtraction will yield more zeros, resulting in better ultimate compression. P- and B-frames are represented in the compressed output as the set of motion estimates and the difference frame. The difference frame is compressed with a JPEG-like DCT compressor.

Motion analysis is very computationally expensive; in fact it is the most computationally expensive component of the MPEG algorithm.

Sequencing of I-, P-, and B-Frames

The series of video frames is therefore encoded by a series of I-, P-, and B-frames. An exemplary pattern could be IBBPBBPBB, and this pattern is then repeated, as shown in Figure 9.

Figure 9
Sequencing I-, P-, and B-frames



RAND *TR216A-9*

Note, however, that to both compress and decompress a B-frame, the subsequent P- or I-frame must be available to the algorithm. This means that the compression and decompression must be done out of order. The actual transmission sequence for the compressed video takes this factor into account, transmitting frames out of order: IPBBPBBIBB, and then the PBBPBBIBB pattern is repeated. This corresponds to transmitting the frames in this sequence: 1, 4, 2, 3, 7, 5, 6, 10, 8, 9, and so on.

Note that if there is a transmission error of an I- or P-frame, subsequent P- and B-frames will propagate this error until the next I-frame. The sequencing of I-, P-, and B-frames can be adapted to the application. Using more P- and B-frames can improve the compression ratios achieved, at the expense of computational complexity and susceptibility to corruption due to transmission error. Using fewer P- and B-frames reduces computational demands and improves the robustness to transmission error, at the expense of reduced compression.

MPEG Summary and Performance Estimates

As mentioned above, I-frames are stand-alone JPEG-like images and are not dependent on any of the prior frames. I-frames thus achieve typical JPEG compression ratios. Typically, P-frames achieve 2–3 times as much compression as the I-frames, and B-frames achieve 2–3 times as much compression as P-frames. Overall, MPEG can achieve compression ratios of more than 200:1, but for high-quality applications the typical range is between 14:1 and 140:1 (Thom and Deutermann, 2001). This large range is a result of variance in how much motion is contained in video data. The interframe similarity and motion analysis is not as effective with high-motion video, resulting in lower compression ratios.

In addition to the video stream, MPEG includes provisions for transmitting synchronized audio, as well as auxiliary data that is synchronized to the video stream (for example, time codes, position information, camera settings, etc.). Audio is usually encoded with a lossy audio compression algorithm, and the other data is usually either transmitted uncompressed (if it is very small) or with lossless compression. In normal applications, the video stream dominates the total amount of data that must be transmitted, and the audio and auxiliary data do not have a great impact on storage size or bandwidth demands.

Summary of Video Compression Algorithm Performance

Table 4 summarizes compression performance estimates for the types of lossless and lossy algorithms discussed in this report.

Without taking advantage of interframe redundancies, any still image compression algorithm (such as gzip) could be applied to video by compressing each individual video frame as if it were a still image. This is the technique used by Motion JPEG 2000, which uses the JPEG 2000 algorithm based on DWT (Yu, 2002). When video compression does not take advantage of interframe similarities, the algorithm's compression ratio for video will be similar to its compression ratio for still images.

In all of the lossy compression algorithms shown, increased compression results in a decrease in quality. Therefore, in practice, the larger compression factors are not useful for the Army's needs. The current standard in the entertainment industry for transmitting high-quality, high-motion video is MPEG-2, using compression ratios near the low end of the compression range (14:1 to 75:1).

When comparing compression ratios of lossy algorithms, it is important to consider the quality factor. For example, two algorithms that achieve a compression factor of 100:1

Table 4
Summary of Compression Performance Estimates

	Compression of Still Images				Compression of Video	
	Lossless		Lossy		Lossy	
Technology	Example Standard	Compression Ratio	Example Standard	Compression Ratio	Example Standard	Compression Ratio
Lempel-Ziv	gzip	1:1 to 9:1				
Predictive/Huffman Coding	Lossless JPEG	1.3:1 to 3:1				
Discrete cosine transform (DCT)			JPEG	3:1 to 100:1	MPEG-2 and MPEG-4	8:1 to 210:1
Discrete wavelet transform (DWT)	JPEG 2000	1.5:1 to 4:1	JPEG	10:1 to 350:1	Motion JPEG 2000	1.5:1 to 350:1

may differ in the quality of the image after decompression. The quality factor is hard to analyze without empirical testing. If the decoded images are to be viewed by humans, then humans should be the judges of quality; but if the decoded images are going to be analyzed by computers, the analysis algorithms should be tested to see how various compression algorithms affect the performance of the analysis algorithms.

Although opinion is not unanimous, it is generally agreed that JPEG 2000 achieves higher quality than JPEG at any given compression ratio. One thing missing from the table is an algorithm that combines the DWT compression of JPEG 2000 with interframe similarity analysis. This may enable compression ratios up to 1000:1 for low-quality video, and ratios of perhaps 45:1 to 250:1 for high-motion, high-quality video. The latter figures assume a level of quality acceptable to applications in the entertainment industry. Even if these hypothetical compression ratios were to be achieved, NTSC video would require 450 kbps of bandwidth and HDTV would require 3.6 Mbps.

Improved throughput of video is attracting increasing attention from commercial industry as firms examine the provision of streaming video to cellular phones. Research at Carnegie Mellon University on advanced H.263 protocols has demonstrated reasonable desktop video data rates at 70 kbps (AMP, 2004). This is achieved by using advanced compression techniques, by reducing frame rates, and by restricting the field of view of the camera. This rate compares favorably with typical commercial fixed-site videoteleconferencing (VTC) data rate requirements of 384 kbps.

CHAPTER SIX

Managing Network Bandwidth

The compression algorithms discussed in this report show promise for reducing the demands for bandwidth when transferring text, images, and video. Reductions of over two orders of magnitude are potentially available. However, there is a loss of quality when lossy compression is used, and the significance of this loss will depend on the needs of the individual user. Some users, at some times, will require the highest level of quality, which will require the most bandwidth. These needs may continually change depending on the operational situation, so prioritizing or predicting what levels of quality will be acceptable must be done dynamically. This chapter discusses some commercially available technologies that enable individual users to tailor their quality of service according to their dynamic needs.

Network Accelerators

Network accelerators are a new technology commonly employed by commercial internet service providers (SlipStream, 2003; Propel, 2004; Radiance, 2004). Network accelerators can improve effective throughput to users by 2–3 times. In addition to improving effective capacity, they allow the user to control the quality of images and potentially the quality of video that is provided. Users are thus able to adjust the relative tradeoff between bandwidth and quality. Moreover, they are able to dynamically modify this tradeoff as events occur.

Network accelerators work by providing more efficient management of internet sessions. Network accelerators use a combination of techniques to improve network capacity:

1. **Compression.** Network accelerators use selectable levels of compression. If the user requests high-quality data, it is passed with lossless or very-high-quality lossy compression. The user can also request other levels of quality, which the system can meet with different levels of compression. If a user wants a large number of images quickly, he can specify high compression to download the images quickly. After review, the user might then select certain images for replacement by higher-quality versions. This enables the user to filter incoming data according to need.
2. **Managing network sessions.** The network accelerator manages the session on the internet to maximize efficiency. For instance, on the World Wide Web, users might find that when downloading a file, the data rate of the download decreases as the session progresses. This occurs because the internet protocol manages the session to balance needs across all users. Network accelerators manage these sessions more efficiently to allow higher data rates for download.

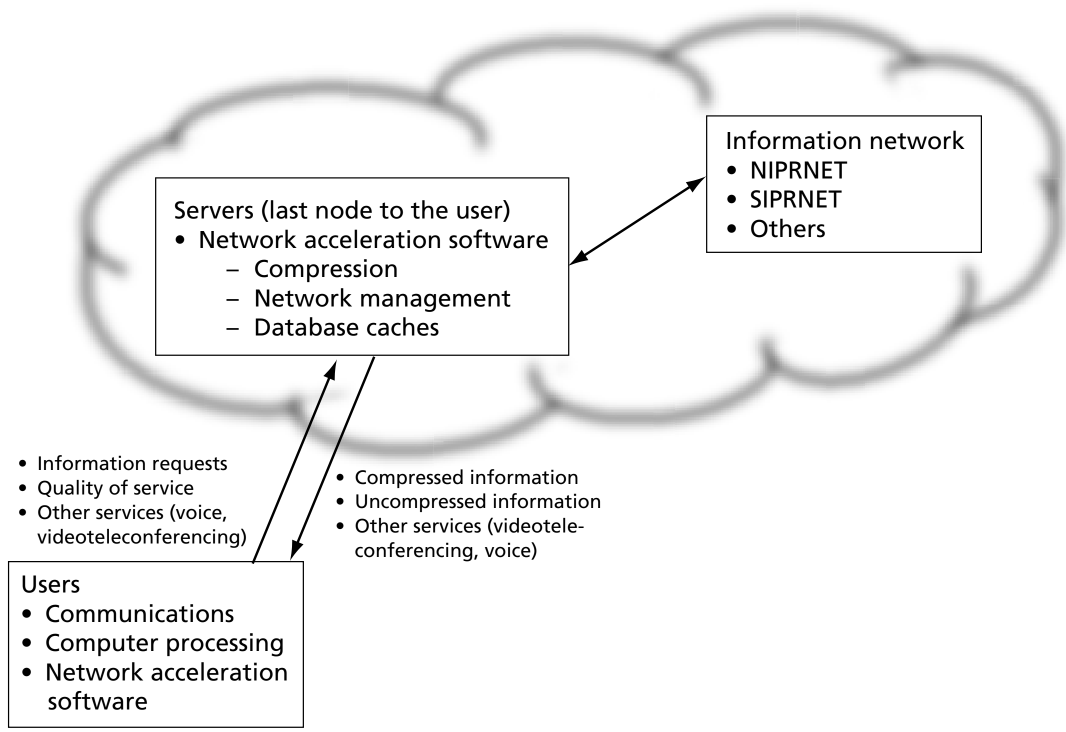
3. **Caching.** Network accelerators will cache data in the network depending on demand. This caching places multiple copies of data within the network. If the data is in high demand, caching reduces the "distance" between the user and the data, thereby reducing the number of intermediate nodes and thus increasing network capacity and reducing transmission delays.

Implementing Network Acclerators

Network accelerators work on the last link in the network connection, between the user and the periphery of the network. The rest of the network is unaffected, as shown in Figure 10. However, this last link, especially to mobile users, is an important one to address because it often has the lowest bandwidth and reliability.

Figure 10 depicts users equipped with network accelerator software requesting information, along with desired quality of service, from the nearest network connection. That connection, equipped with network accelerator software and possibly hardware, then obtains the desired information from the larger network, applies the desired quality of service (e.g., level of compression), and transmits the data to the user. The network accelerator thus requires installation only at the user level and at the first link into the larger internet.

Figure 10
How Network Accelerators Overlay the Existing Network



RAND *TR216A-10*

How the Army Can Take Advantage of Network Accelerators

As discussed above, implementation of network accelerators requires modification to the user-level systems and to the network's point-of-entry systems. The software on the user's computer then negotiates with the nearest server, which obtains the information, caches the original data, applies compression to achieve the desired quality of service, and transmits the data to the user. This then improves the capacity of the last link, or "last mile," connection and leaves the rest of the data on the network untouched. Desired level of quality for an individual user only affects the information passed along that user's last mile link, which is often the slowest link and thus will benefit the most from compression.

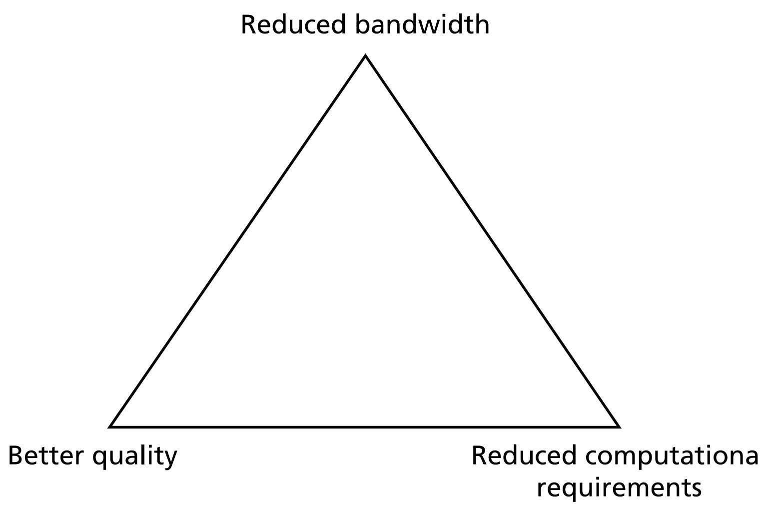
Network acceleration can improve capacity by factors of 2 to 3 depending on use. The Army could take advantage of advanced compression techniques, as discussed earlier in this report, to improve network performance. Current commercial systems focus on compression of imagery and do not currently accelerate real-time streaming connections used for voice and video, which will be useful for distributed videoteleconferences. Applying network acceleration technology to this type of data will require some investment by the Army in building upon the commercial technologies.

CHAPTER SEVEN

Concluding Remarks

The issues and tradeoffs in the solution space of video formats and compression schemes are summarized in Figure 11. These broad categories encompass several of the considerations detailed in this report.

Figure 11
Required Tradeoffs



RAND *TR216A-11*

For example, bandwidth requirements affect network capacity requirements and the cost of acquiring that capacity, and may impact security and the reliable delivery of data; quality includes measures such as image fidelity and latency; and computational demands affect equipment cost, power demand, and mobility. Increases or decreases in any of these broad quantities will have an impact on at least one of the other quantities. Here are some examples:

- In general, as compression is increased, the quality of the received video stream is decreased. Whether humans or automated systems will interpret the video, reduction in video quality or fidelity can hinder analysis. However, there are currently no objective standards for measuring or assessing the quality of the compressed output.
- Compression algorithms that attempt to retain quality while using high compression ratios will often require significant processing at the source or destination ends. Processing resources may be limited at the sources of these video streams, such as UAVs,

as well as in some types of receiving nodes in the field (OSD, 2002). Increased processing demands can also increase demands on batteries or other power sources. Variations in algorithms can sometimes shift these burdens among the source, intermediate, and destination nodes.

- When the software processing demands of compression or decompression algorithms are high, specialized hardware can often be developed to replace software algorithms. This can increase speed but may also increase equipment costs. As mentioned earlier, economies of scale can lower equipment costs if consumers are also using the same technologies, as is the case with JPEG and MPEG.
- As compression is increased and computational demands increase, latencies can be induced into the video stream. In some cases, latencies may be unacceptable, such as when remotely controlling an unmanned vehicle. In addition, if compression cannot happen in real time, the source node may have to temporarily store uncompressed data for future transmission, imposing larger delays and additional hardware requirements.

Even beyond compression, these three main attributes impose tradeoffs. For example, commercial network management technologies such as EarthLink's Propel (Propel, 2004) supplement compression with additional methods to address bandwidth constraints. They reduce the amount of overhead in setting up and breaking down connections by keeping connections open longer and reusing them; they cache data near end users so that it can be downloaded with lower latency than if it had to be retrieved over the internet at large; and they apply a user-tunable amount of compression to web images to reduce the data that must be transmitted over the slowest portion of the link, the last mile to the user's home. Each of these, of course, imposes tradeoffs on quality and/or computational demands (e.g., for network management and storage).

Summary and Recommendations for the Army

- The Army can take advantage of commercial compression technologies. However,
  - Acceptable levels of quality must be determined.
  - User equipment may require increased computing power.
- Network accelerators can improve throughput by factors of 2 to 3. However, they require
  - Changes to network structure and operations, but only for the last mile to the user.
  - Research and development into techniques to enhance real-time streaming data.
  - User training in managing demands for quality.
- We recommend that the Army
  - Incorporate compression and network acceleration technologies into future systems.
  - Identify where Army-specific tailoring could improve on commercial technologies.
  - Develop an experimental plan to determine acceptable levels of quality and to train users.

References

3ID, *Third Infantry Division (Mechanized) After Action Report, Operation Iraqi Freedom*, July 2003.

AMP, *About the Advanced Multimedia Processing Lab*, Carnegie Mellon University, 2004. http://amp.ece.cmu.edu/about.htm.

Balogh, Nandor, Vytenis Punys, Jurate Puniene, Jonas Punys, and Vytautas Vaitkevicius, *Recommendations on the Use of Lossy Compression in Clinical Environment* [sic], INCO-COPERNICUS Project No. PL961144-SAMTA, 2000.

Carslaw, H.S., *Introduction to the Theory of Fourier's Series and Integrals,* New York: Dover Publications, 1952.

CBO, *The Army's Bandwidth Bottleneck*, Congressional Budget Office, August 2003.

CECOM, *Multifunctional On-the-Move Secure Adaptive Integrated Communications (MOSAIC) ATD*, briefing, U.S. Army CECOM RDEC, March 2002.

Christopoulos, Charilos, Athanassios Skodras, and Touradj Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview," *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 4, November 2000, pp. 1103–1127.

GIG, *Global Information Grid Overarching Policy*, Department of Defense Directive 8100.1, September 19, 2002.

Hillman, J., S. Jones, R. Nichols, and I. Wang, "Communications Network Architectures for the Army Future Combat System and Objective Force," IEEE MILCOM, 2002.

JHU, *Network-Centric Architectures for the Army Future Combat Systems and Objective Force Phase II: Architecture Performance Assessment,* Laurel, MD: Johns Hopkins University Applied Physics Laboratory, JWR-02-004, February 2002.

Joe, Leland, and Isaac Porche, *Future Army Bandwidth Needs and Capabilities*, Santa Monica, CA: RAND Corporation, MG-156-A, 2004.

JPEG, *JPEG 2000—Links to Information*, Joint Photographic Experts Group, 2003. http://www.jpeg.org/JPEG2000.html.

Kosheleva, Olga, Carlos Mendoza, and Sergio D. Cabrera, "Task-Specific Image Quality Metrics for Lossy Compression of FLIR Images," Proceedings of SPIE, 1999. http://citeseer.nj.nec.com/235231.html.

Lai, Yung-Kai, Jin Li, and C.-C. Jay Kuo, "A Wavelet Approach to Compressed Image Quality Measurement," IEEE 30th Annual Asilomar Conference on Signals, Systems, and Computers, 1996. http://citeseer.nj.nec.com/lai96wavelet.html.

Moran, Dennis, *Statement by BG(P) Dennis Moran, Director, Information Operations, Networks and Space, CIO/G-6, US Army*, before the Subcommittee on Terrorism, Unconventional Threats and Capabilities, Armed Services Committee, U.S. House of Representatives, October 21, 2003.

MPEG, *The MPEG Home Page*, Moving Picture Experts Group, 2003. http://www.chiariglione.org/mpeg/index.htm.

NATO, *The NATO Post-2000 Narrow Band Voice Coder: Test and Selection of STANAG 4591*, NATO C3 Agency, Technical Presentation TP-001, 2002.

OSD, *Unmanned Aerial Vehicles Roadmap: 2002–2027*, Office of the Secretary of Defense, December 2002.

Propel, *Propel Accelerator: How It Works*, 2004. http://www.propel.com/ac/howitworks.jsp.

Radiance, *U.S. Army Relies on Radiance TrueDelivery System to Improve Data Delivery to and from the Warfighter Edge in Operation Iraqi Freedom*, Radiance Technologies, Inc., September 2004.

Raduege, Harry, *Statement by LTG Harry Raduege, Jr.*, before the Subcommittee on Terrorism, Unconventional Threats and Capabilities, Armed Services Committee, U.S. House of Representatives, April 3, 2003.

———, *Net-Centricity: The Core of DoD Transformation*, briefing by LTG Harry Raduege, Jr., Director of DISA, February 19, 2004.

Rountree, Janet C., Brian N. Webb, and Michael W. Marcellin, "Testing JPEG 2000 Compression for Earth Science Data," Earth Science Technology Conference, NASA, Pasadena, CA, 2002. http://esto.gsfc.nasa.gov/conferences/estc-2002/Papers/A3P1(Rountree).pdf.

Shaaber, Major Mark, Captain Scott Hedberg, and Mr. Troy Wesson, *V Corps: C4ISR Integration AAR, A Paper on the Information Systems Integration and Information Management Challenges and Successes Training for and Achieving "Victory" in Operation Iraqi Freedom*, May 2003.

SlipStream, *Optimizing Data Communication with the SlipStream (TM) SP Acceleration Platform*, SlipStream Data, Inc., November 2003.

Thom, Gary A., and Alan R. Deutermann, *A Comparison of Video Compression Algorithms*, Delta Information Systems, 2001. http://www.delta-info.com/DDV/downloads/Comparison%20White%20Paper.pdf.

USMC, *Information Management Issues Emerging from USMC Experience in Operation Iraqi Freedom*, Enduring Freedom Combat Assessment Team, USMC, briefing to MORS, October 28, 2003.

Volder, Jack E., "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computers,* Vol. EC-8, September 1959, pp. 330–334.

Wallace, William, *Statement by LTG William Wallace, CG, CAC, US Army TRADOC*, before the Subcommittee on Terrorism, Unconventional Threats and Capabilities, Armed Services Committee, U.S. House of Representatives, October 21, 2003.

Yeh, Pen-Shu, and Jack Venbrux, *A High Performance Image Data Compression Technique for Space Applications*, Greenbelt, MD: Goddard Space Flight Center, 2002.

Yu, Wei, *Motion JPEG 2000 and Wavelet-Based Coding in Video and Image Processing*, master's thesis, St. Louis, MO: Department of Computer Science and Engineering, Washington University, 2002. http://citeseer.nj.nec.com/yu02motion.html.

Ziv, J., and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory,* Vol. 23, 1977, pp. 337–342.